云南大学数学系《运筹学通论实验》课程上机实验报告

课程名称：运筹学通论	**学期**：2015-2016 学年第二学期	**成绩**：
指导教师：李建平	**学生姓名**：金洋	**学生学号**：20131910023
实验名称： **Prim**		
实验编号：No.5	**实验日期**： 2016/5/27	**实验学时**： **1**
学院： 数学与统计学院	**专业**： **信息与计算科学**	**年级**： **2013**

一、实验目的

使用 c 语言实现 Prim 算法(反圈法)，并用其解决

1.最小支撑树问题（MST）；

2.求一个有向图中所有最短路(S-P)构成的子图；

二、实验内容

掌握 Prim 算法，使用 C 语言实现该算法，并用其解决最小支撑树问题、求一个有向图中所有最短路构成的子图；

三、使用环境

平台：Microsoft Visual C++ 6.0

语言：C 语言

四、算法介绍

1.MST

Algorithm Prim(MST)

Input: G=(V,E);

Output: a minimal spanning tree of G or "G is not connected.";

Begin

Step1: X={v_1}, F=∅ (F 是边集)；

Step2: while (X≠V and $\Phi(X)\neq\emptyset$) do

1.choose an edge e=xy∈ $\Phi(X)$ which has the minimum

weight;(Φ (X)={xy∈E|x∈X, y∉X})

2.X=X∪{y};

3.F=F∪{xy};

Step3: if (X=V) then output MST T=(V,F)

else output "G is not connected.";

End.

2. S-P

Algorithm Prim(S-P)

Input: D=(V,A), v_s, v_t;

Output: 从v_s到v_t的所有最短路径构成的子图，或v_s到v_t不存在最短路;

Begin

Step1: X={ v_s }, $\lambda(v_s)=0$, $l(v_s)=0$, $\lambda(v_i)=0$ (i≠s);

Step2: while ($v_t \notin X$ and Φ^+(X)≠∅) do

1.在Φ^+(X)中选取所有满足条件的弧, $e_{i_j}=(u_{i_j},v_{i_j})\in\Phi^+(X)$，满足 $\lambda(u_{i_j})+w(u_{i_j},v_{i_j})=\min\{\lambda(u')+w(u',v')\,|\,(u',v')\in\Phi^+(X)\}$,j=1,2,...k;

2. X=X∪{ $v_{i_j}\,|\,j=1,2,\cdots,k$ }

3. F=F∪{ $(u_{i_1},v_{i_j})\,|\,j=1,2,\cdots,k$ };

4. $l(v_{i_j})=\{u_{i_j}\}, j=1,2,\cdots,k$

Step3: if ($v_t\in X$) then 输出从v_s到v_t的所有最短路径构成的子图;

else 输出“从v_s到v_t不存在最短路”;

End.

五、调试过程

1．程序代码

①MST.c

```
#include <stdio.h>
#define MAXNUM 999999999
#define MAXVERTEX 100

int
M[MAXVERTEX][MAXVERTEX],X[MAXVERTEX],F[2][MAXVERTEX],n,Fnu
m,SUM;

/*X 是否等于 V     是:返回 1;否:返回 0*/
int equal() {
    //X[0]记录 x 集合中顶点个数
    return X[0]==n;
}

/* ϕ(X)是否为空?   是:返回 0;否:返回 1*/
int PHi() {
    int i,j;
    for(i=1;i<=n;i++)
        for (j=1;j<=n;j++)
            if ((i!=j)&&(X[i])&&(X[j]==0)&&(M[i][j]<MAXNUM)) return(1);
    return(0);
}
```

```
void prim() {
    int i,j,min,x=0,y=0;
    min=MAXNUM;
    for(i=1;i<=n;i++)
        if (X[i])

        for(j=1;j<=n;j++)
            if ((!X[j])&&(i!=j))
                if (M[i][j]<min) {
                    min=M[i][j];
                    x=i;
                    y=j;
                }
    if (min!=MAXNUM) {
        X[y]=1;
        X[0]++;
        Fnum++;
        F[0][Fnum]=x;
        F[1][Fnum]=y;
        SUM=SUM+min;
    }
}

void main() {
    int i,j;
```

```
    //freopen("MST.in", "r", stdin);
    printf("请输入顶点个数 n=");
    scanf("%d",&n);
    printf("请输入邻接矩阵:\n");
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++) {
            scanf("%d",&M[i][j]);
            if (M[i][j]==0) M[i][j]=MAXNUM;//没有边相连，将权值设为足够大
        }

    X[1]=1;
    X[0]=1;//X[0]记录 x 集合中顶点个数
    for (i=2;i<=n;i++) X[i]=0;

    while (!equal() && PHi()) prim();

    if (equal()) {
        printf("MST 的边集为:\n");
        for(i=1;i<=Fnum;i++)
            printf("[%d,%d]\n",F[0][i],F[1][i]);//输出 MST 的边集合
        printf("The total weight is %d.\n",SUM);
    }
    else printf("G is not connected.\n");
}
```

② S-P.c

```
#include <stdio.h>
#define MAXNUM 999999999
#define MAXVERTEX 100

int
M[MAXVERTEX][MAXVERTEX],X[MAXVERTEX],LAM[MAXVERTEX],L[MAXVERTEX][MAXVERTEX],n,VS,VT;


/* φ(X)是否为空?  是:返回 0;否:返回 1*/
int PHi() {
    int i,j;
    for(i=1;i<=n;i++)
        for (j=1;j<=n;j++)
            if ((i!=j)&&(X[i])&&(X[j]==0)&&(M[i][j]<MAXNUM)) return(1);
    return(0);
}

void prim() {
    int i,j,min;
    int now[MAXVERTEX];//表示某顶点是否在这一轮反圈中选边时被加入X集合;

    for (i=1;i<=n;i++) now[i]=0;
    min=MAXNUM;
```

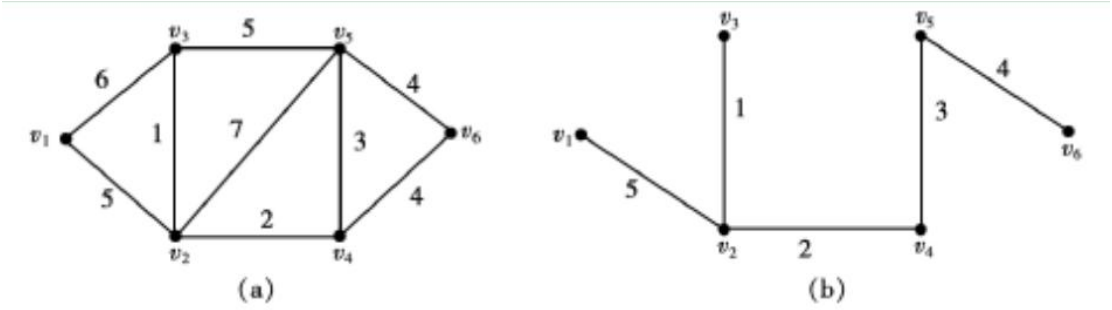
```
    for(i=1;i<=n;i++)
        if (X[i])
            for(j=1;j<=n;j++)
                if (!X[j]    && i!=j && LAM[i]+M[i][j]<min )
                    min=LAM[i]+M[i][j];

    if (min!=MAXNUM) {
        for(i=1;i<=n;i++)
            if (X[i])
                for(j=1;j<=n;j++)
                    if ( (!X[j] || now[j]) && i!=j && LAM[i]+M[i][j]==min ) {
                        LAM[j]=min;
                        X[j]=1;
                        now[j]=1;
                        //X[0]++;
                        L[j][i]=1;//表示弧(i,j)有可能在最终形成的最短路上

                    }
    }
}


int display(int v) {
    int i;
    if (v==VS) return 0;
```

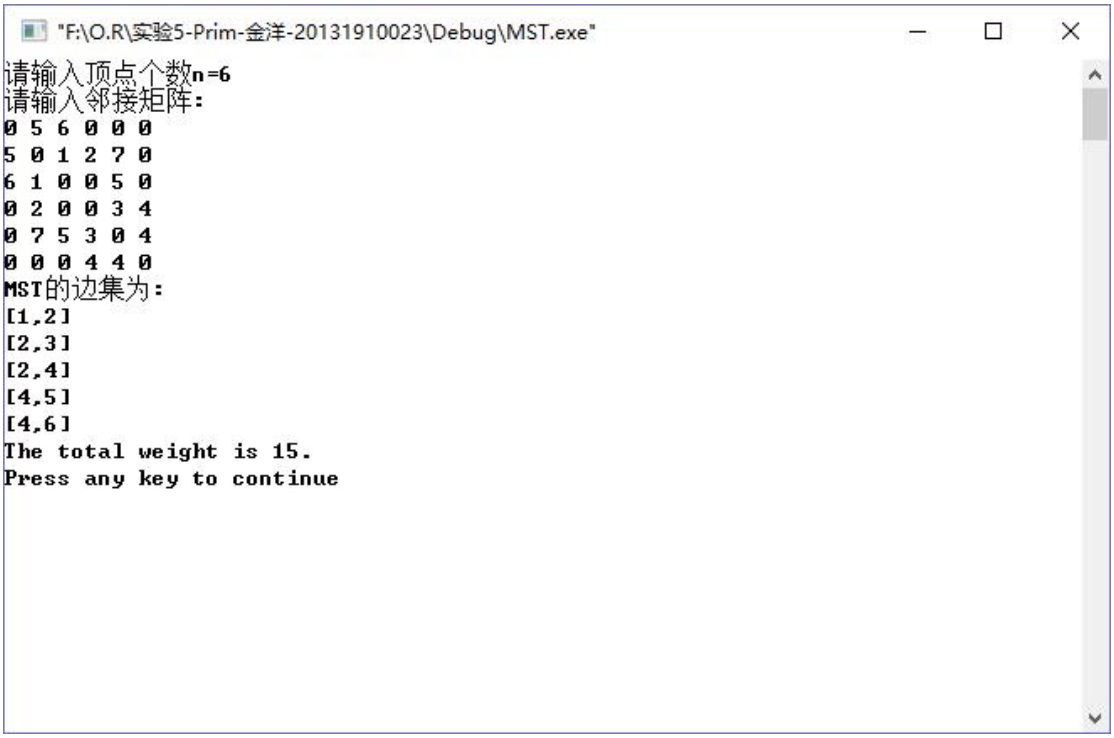
```
    for (i=1;i<=n;i++)
        if (L[v][i]) {
            display(i);
            printf("(%d,%d)\n",i,v);
        }
    return 0;
}



void main() {
    int i,j;
    //freopen("S-P.in", "r", stdin);
    printf("请输入顶点个数 n=");
    scanf("%d",&n);
    printf("请输入有向图的邻接矩阵:\n");
    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++) {
            scanf("%d",&M[i][j]);
            if (M[i][j]==0) M[i][j]=MAXNUM;//没有边相连，将权值设为足够大
        }
    printf("起点 Vs=");
    scanf("%d",&VS);
    printf("终点 Vt=");
    scanf("%d",&VT);
```

```
    //X[0]=1;//X[0]记录 X 集合中顶点个数
    for (i=1;i<=n;i++) {
        X[i]=0;
        LAM[i]=MAXNUM;
    }
    X[VS]=1;
    LAM[VS]=0;

    for (i=1;i<=n;i++)
        for (j=1;j<=n;j++)
            L[i][j]=0;

    while (!X[VT] && PHi()) prim();

    if (X[VT]) {
        printf("所有最短路径构成的子图的边集为:\n");
        display(VT);
        printf("最短路径长为%d.\n",LAM[VT]);
    }
    else printf("从 Vs 到 Vt 不存在最短路.\n");
}
```

2．运行窗口

①MST　example 课本（第三版）P261



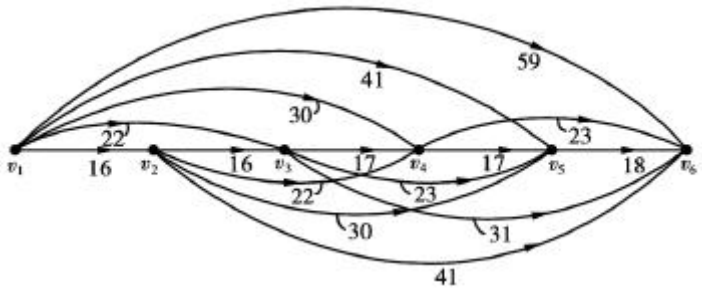



"F:\O.R\实验5-Prim-金洋-20131910023\Debug\MST.exe"
请输入顶点个数n=6
请输入邻接矩阵:
0 0 0 0 0 0
0 0 1 2 7 0
0 1 0 0 5 0
0 2 0 0 3 4
0 7 5 3 0 4
0 0 0 4 4 0
G is not connected.
Press any key to continue

② S-P


"F:\O.R\实验5-Prim-金洋-20131910023\Debug\S-P.exe"
请输入顶点个数n=6
请输入有向图的邻接矩阵:
0 0 0 0 0 0
0 0 1 2 7 0
0 1 0 0 5 0
0 2 0 0 3 4
0 7 5 3 0 4
0 0 0 4 4 0
起点Vs=1
终点Vt=6
从Vs到Vt不存在最短路.
Press any key to continue

课本例 13




```
"F:\O.R\实验5-Prim-金洋-20131910023\Debug\S-P.exe"
请输入顶点个数n=6
请输入有向图的邻接矩阵:
0 16 22 30 41 59
0 0 16 22 30 41
0 0 0 17 23 31
0 0 0 0 17 23
0 0 0 0 0 18
0 0 0 0 0 0
起点Vs=1
终点Vt=6
所有最短路径构成的子图的边集为:
<1,3>
<3,6>
<1,4>
<4,6>
最短路径长为53.
Press any key to continue
```

六、总结

1. 掌握了 Prim 算法，学会使用 C 语言实现该算法，并用其解决最小支撑树问题、求一个有向图中所有最短路构成的子图；

2. 针对不同问题，要学会思考，是否可以使用一个思想、算法，稍加修改来使其适用不同的问题。

3.示例的问题规模比较小，采用数组数据结构已经适用，如果问题规模增大，可以考虑使用链表。

七、参考文献

[1] 谭浩强著, 《c 程序设计》（第三版）, 清华大学出版社, 2005.7；
[2] 《运筹学》教程编写组, 《运筹学》（第 4 版）, 清华大学出版社, 2013.1;

八、教师评语